Scholia 2026: Compliance with SPARQL 1.1

Egon L. Willighagen[1], Daniel Mietchen[2,3,4], Peter Patel-Schneider, Konrad Linden[5], Lars G. Willighagen[6] and Wolfgang Fahl[7]

[1]*Department of Translational Genomics, NUTRIM, Maastricht University, The Netherlands*
[2]*FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Karlsruhe, Germany*
[3]*Leibniz Institute of Freshwater Ecology and Inland Fisheries (IGB), Berlin, Germany*
[4]*Institute for Globally Distributed Open Research and Education (IGDORE), Jena, Germany*
[5]*Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany*
[6]*Department of Ecology, Radboud Institute for Biological and Environmental Sciences, Radboud University, The Netherlands*
[7]*BITPlan GmbH, Willich, Germany*

Abstract

If Hannah is listed as an author, her mentions in the text and in the acknowledgments should be adjusted accordingly Scholia is a specialized portal that uses a combination of SPARQL and the Flask Python platform to visualize data from Wikidata. In this demonstration, we will show how Scholia works and how the Scholia team made it independent from the Blazegraph-based Wikidata Query Service (WDQS) over the past eighteen months. This had become necessary due to a forced Wikidata graph split in 2025, which affected Scholia because most of its SPARQL queries were tuned towards the WDQS dialect of SPARQL. The project explored various options, including writing federated SPARQL queries using the new functionalities provided by the split WDQS, but here, we discuss a concerted effort to make Scholia queries compatible with a SPARQL 1.1 compliant engine, QLever.

Keywords

Scholia, Wikidata, SPARQL, Python

Wikidata is a crowd-curated platform providing general reference information in a semantic format [1]. Its official SPARQL service, the Wikidata Query Service (WDQS), has historically relied on the Blazegraph engine to evaluate queries [2]. However, as Wikidata grew to over 17 billion triples, its Blazegraph backend has reached significant performance limits. This led the Wikimedia Foundation to split the query service into two parts in 2025: WDQS-main, containing roughly 8.6 billion triples of general data, and WDQS-scholarly, containing 8.7 billion triples related to scholarly articles [3].

Scholia [4] is a specialized portal for the scientific community that serves Wikidata content through a set of 387 precomposed SPARQL query templates organized into over 40 profile types, known as aspects. These cover general entities such as authors, works, venues and organizations as well as more special ones like taxa, diseases, chemicals, genes or proteins [5]. Because Scholia's queries frequently require data from both the general and scholarly Wikidata subsets, the split posed a critical challenge to the continuation of the service. As an interim solution, the Wikimedia Foundation continued to provide a Blazegraph instance with the unified full graph until 20 January 2026. Preparing for this change, the Scholia team investigated three primary recovery routes, ultimately selecting the third one:

1. Modifying existing queries to use SPARQL federation to communicate between the two split endpoints. This was found to be impractical, as inter-service communication led to severe performance degradation and the loss of optimization opportunities [3].
2. Using snapQuery middleware [6] to generalize query access across endpoints.
3. Remove any Blazegraph-specific constructs in Scholia and instead changing all the queries to standard SPARQL 1.1. This allows the use of any SPARQL 1.1 engine that can effectively evaluate queries against the entire Wikidata graph, including QLever [7], MilleniumDB, and Virtuoso, not

SWAT4HCLS 2026: The 17th International Conference on Semantic Web Applications and Tools for Health Care and Life Sciences, March 23–26, 2026, Amsterdam, The Netherlands
egon.willighagen@maastrichtuniversity.nl (E. L. Willighagen)
0000-0001-7542-0286 (E. L. Willighagen); 0000-0001-9488-1870 (D. Mietchen); 0009-0000-7789-7459 (P. Patel-Schneider); 0009-0009-1449-5423 (K. Linden); 0000-0002-4751-4637 (L. G. Willighagen); 0000-0002-0821-6995 (W. Fahl)

only obviating the need to switch to a split-service architecture with its inherent inefficiencies but also leveraging modern engines that are considerably faster than Blazegraph.

A fork of the original Scholia project was created [8] to collaborate on the necessary modifications, around which several hackathons were organized [9]. A decision was made to use a Wikidata query service (https://qlever.dev/api/wikidata) powered by QLever [7]. This decision was supported by benchmarking multiple SPARQL backends [10], of which QLever was fastest overall.

The bulk of the work was eliminating non-standard constructs from query templates. The major types of these constructs were: named sub-queries; the Blazegraph locality service that finds nodes with a geographic property value near a given geographic point; the Blazegraph gather-scatter service that finds nodes in the vicinity of a node reachable by a property; the Wikibase service allowing communication with the MediaWiki API for e.g. searching full text contents of Wikipedias; and the Wikibase label service that finds the property value that has the most-preferred language. These non-standard constructs had largely been used to improve query evaluation. Other systems using the WDQS also use non-standard constructs to get around problems with query termination caused by the slowness of Blazegraph.

Named sub-queries were eliminated by substituting the named query where it was referenced. The locality service was eliminated by unrestricted querying and then filtering based on computed distances. The gather-scatter service was mostly eliminated by using transitive closure and filters or limits. Perhaps the most problematic construct to eliminate was the Wikibase label service, as it involves preference. Based on suggestions from Hannah Bast, the leader of the QLever team, Jinja templates were implemented that generate standard SPARQL 1.1 constructs to efficiently determine the best labels or descriptions. The templates take a Wikidata item plus a sequence of language codes and the generated SPARQL effectively iterates over the language codes, and returns the label with the first language code for which the item has a label or description in that language code. Versions of the templates had to be written to work well with unbound query variables, as can arise in queries with `OPTIONAL`.

In general, though the process took about eighteen months, the overall results are quite satisfactory, and the QLever-based setup exhibits improved performance compared to the original Blazegraph one. While the official Scholia service is yet to be updated to use the new QLever-based setup, our effort shows that it is generally possible to adjust any system that uses the WDQS to instead use standard SPARQL 1.1 queries.

The Wikimedia Foundation is looking into making its official Wikidata Query Service SPARQL 1.1. compliant, too. If, contrary to current expectations, that service does not use QLever, the Scholia queries may need some further adjustments.

The demo at SWAT4HCLS will be set up to demonstrate Scholia, along with opportunities to investigate the changes made in the conversion and effects of using QLever.

Acknowledgments

The authors thank everyone who has contributed patches or bug reports to the Scholia project, particularly Finn Nielsen (for the original concept of Scholia), Hannah Bast, Moritz Schubotz and Lane Rasberry. A rich list of contributors to the project is available at https://github.com/WDscholia/scholia/graphs/contributors. The work described here was supported in part by the German Research Foundation through the find.software project [11] (funded under grant number 567156310) and by the European Commission through the AQUANAVI initiative [12] of the OSCARS project (101129751).

Declaration on Generative AI

Some author(s) have occasionally employed Generative AI tools for the changes in the Scholia code. All code changes have been reviewed by another person.

References

[1] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85. doi:10.1145/2629489.

[2] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, A. Bielefeldt, Getting the most out of Wikidata: Semantic technology usage in Wikipedia's knowledge graph, in: International Semantic Web Conference, Springer, 2018, pp. 376–394.

[3] T. Lubiana, L. Rasberry, D. Mietchen, The Wikidata Query Service Split and its Impact on the Scholarly Graph, in: CEUR Workshop Proceedings, volume 4064, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-4064/PD-paper3.pdf.

[4] F. Å. Nielsen, D. Mietchen, E. Willighagen, Scholia, Scientometrics and Wikidata, in: The Semantic Web: ESWC 2017 Satellite Events, ESWC 2017 Satellite Events, Springer International Publishing, Portorož, SI, 2017, pp. 237–259. doi:10.1007/978-3-319-70407-4_36.

[5] E. Willighagen, D. Slenter, A. Rutz, D. Mietchen, F. Å. Nielsen, Scholia Chemistry: access to chemistry in Wikidata (2025). doi:10.26434/CHEMRXIV-2025-53N0W.

[6] W. Fahl, T. Holzheim, C. T. Hoyt, D. Priskorn, E. Willighagen, SnapQuery, 2025. URL: https://github.com/WolfgangFahl/snapquery.

[7] H. Bast, B. Buchhold, QLever: A Query Engine for Efficient SPARQL+Text Search, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 647–656. doi:10.1145/3132847.3132921.

[8] F. Å. Nielsen, D. Mietchen, E. Willighagen, H. Bast, Scholia, 2017. URL: https://github.com/ad-freiburg/scholia/tree/qlever, GitHub repository qlever branch.

[9] Wikidata, Scholia Events December 2025, 2025. URL: https://www.wikidata.org/wiki/Wikidata:Scholia/Events/2025_12.

[10] P. F. Patel-Schneider, Benchmarking SPARQL Engines on Wikidata Queries, in: CEUR Workshop Proceedings, volume 4108, CEUR-WS.org, 2024. URL: https://ceur-ws.org/Vol-4108/paper3.pdf.

[11] R. Gey, D. Mietchen, O. Karras, T. Wittenborg, M. Schubotz, J. Bumberger, find.software: Foundations for Interdisciplinary Discovery of (Research) Software, Research Ideas and Outcomes 11 (2025) e179253. doi:10.3897/rio.11.e179253.

[12] T. Heger, S. Angela Berger, J. M. Jeschke, C. Kittel, P. Kraker, A. K. Makower, D. Mietchen, J. C. Nejstgaard, M. Schramm, AQUANAVI: Navigating Grand Challenges and their Mitigation using Aquatic Experimental RIs, Research Ideas and Outcomes 11 (2025) e176476. doi:10.3897/rio.11.e176476.